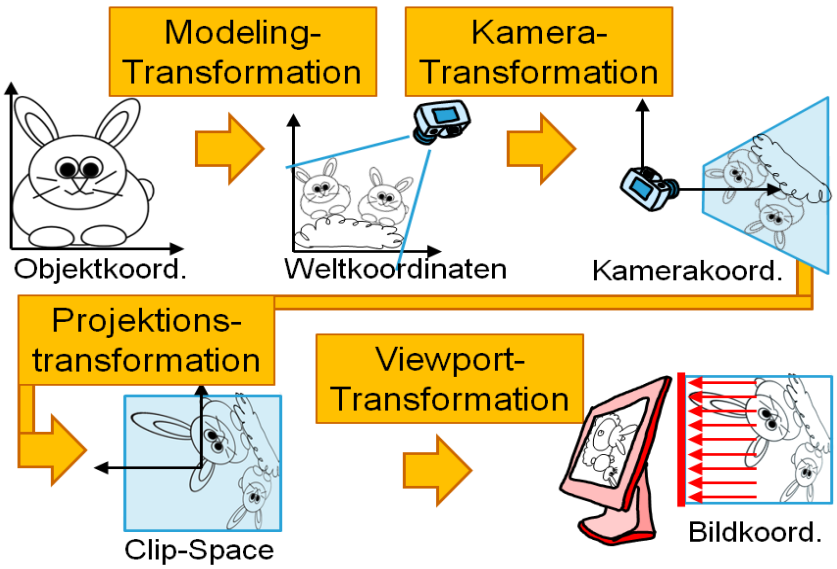
7. Viewing

Viewing in der Graphik-Pipeline

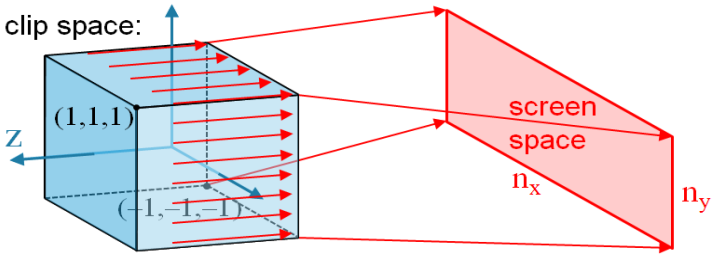
Grundsätzlich erfolgt die Modellierung der darzustellenden Szene in Weltkoordinaten, wobei einzelne Teile aus deren lokalen Koordinatensystemen mit geometrischen Transformationen (Matrizen!) in Weltkoordinaten umgewandelt werden. Nach der Definition der Kameraparameter werden die Koordinaten in Kamerakoordinaten verwandelt, auf die anschließend die Projektion angewandt wird. Das Ergebnis liegt in einem normierten Würfel (häufig der Seitenlänge 2), von wo es durch die sogenannte Viewport-Transformation in die Gerätekoordinaten des jeweils verwendeten Ausgabemediums transformiert wird.



In der Geometrie sind mehrere verschiedene Projektionen bekannt, von denen in der Computergraphik aber nur die Parallelprojektion und die Perspektive eine größere Bedeutung haben. Wir werden nun zuerst annehmen, dass die Kamera eine Parallelprojektion ausführt, und anschließend überlegen, wie man auch die perspektivische Projektion in die Pipeline einbauen kann. Dabei fangen wir von hinten an, weil es so einfacher zu überlegen ist.

Viewport-Transformation

Wir nehmen also an, dass die Szene bereits im Clip-Space vorhanden ist, d.h. alle relevanten Koordinaten befinden sich in einem achsenparallelen Würfel der Seitenlänge 2 mit Mittelpunkt (0,0,0). Wir wollen eine orthographische Abbildung (parallele Normalprojektion) mit Blickrichtung –z auf einen Bildschirm mit Abmessungen $n_x \times n_y$ (Pixel) durchführen. Es müssen also alle Punkte (–1,–1,z) auf (0,0) abgebildet werden und alle Punkte (1,1,z) auf (n_x,n_y). Diese lineare Abbildung wird durch die Matrix M_{vp} bewerkstelligt:

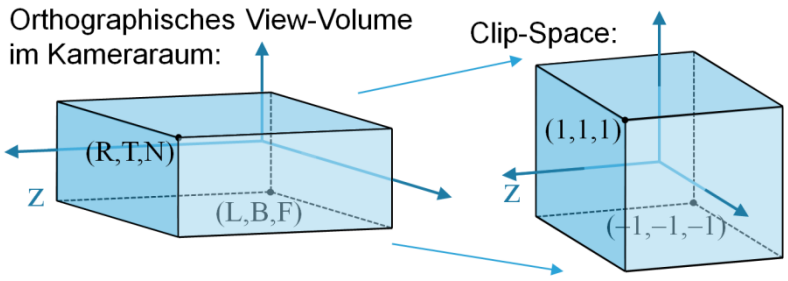


$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ z \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} n_x/2 & 0 & 0 & n_x/2 \\ 0 & n_y/2 & 0 & n_y/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M_{vp}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Deren Richtigkeit kann sofort nachgewiesen werden, indem man die Eckpunkte einsetzt. Die Matrix weist aber in den roten Zahlen noch eine Besonderheit auf: die z-Werte werden erhalten! Dies ist im Moment ohne Belang, wird aber bei späteren Schritten (vor allem bei der Berechnung der Sichtbarkeit) noch von großem Wert sein.

Projektionstransformation

Wie bereits oben erwähnt, nehmen wir für den Moment an, dass eine orthographische Projektion durchzuführen ist. Das vereinfacht die Transformation gewaltig, denn jetzt müssen wir nur einen achsenparallelen Quader mit den Grenzen L(eft), R(ight), B(ottom), T(op), N(ear), F(ar) so verschieben und verzerren, dass der Würfel $[-1,-1]^3$ entsteht (siehe Abb.). (L,B,F) muss also zu (–1,–1,–1) werden und (R,T,N) zu (1,1,1). Auch dies lässt sich wieder einfach mit einer Transformationsmatrix ausdrücken:



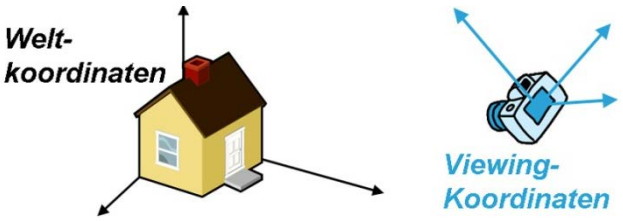
$$M_{orth} = \begin{bmatrix} \frac{2}{R-L} & 0 & 0 & -\frac{R+L}{R-L} \\ 0 & \frac{2}{T-B} & 0 & -\frac{T+B}{T-B} \\ 0 & 0 & \frac{2}{N-F} & -\frac{N+F}{N-F} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Bemerkung: Eine Parallelprojektion kann auch schräg auf eine Abbildungsebene erfolgen (zum Beispiel beim Schattenwurf), diese Variante wird hier nicht berücksichtigt.

Kamera-Transformation

Ähnlich wie beim Fotografieren hat man beim Festlegen der Kamerawerte mehrere Freiheitsgrade:

1. Position der Kamera im Raum
2. Blickrichtung von dieser Position aus
3. Orientierung der Kamera (wo ist oben?)
4. Größe des Bildausschnittes (entspricht der Brennweite bzw. dem Zoomfaktor bei einem Fotoapparat)



Aus den ersten drei Werten berechnet man das *Kamerakoordinatensystem u,v,w* (Viewing-Koordinaten). Normalerweise ist die uv-Ebene dieses Viewing-Koordinatensystems normal auf die Hauptblickrichtung, und man *blickt in die Richtung der negativen w-Achse*.

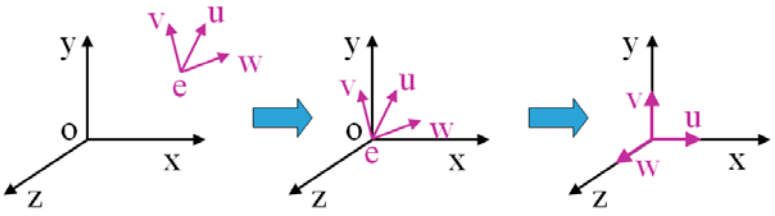
Ausgehend von der Kameraposition geht man prinzipiell folgendermaßen vor, um das Viewing-Koordinatensystem festzulegen:

1. Wahl einer Kameraposition (auch Augpunkt oder Viewing-Punkt genannt).
2. Wahl einer Blickrichtung, die negative Blickrichtung ergibt die w-Achse.
3. Wahl einer Richtung t „nach oben“; aus dieser lassen sich dann die u- und v-Achsen berechnen.
4. Da die Abbildungsebene normal auf die Blickrichtung liegt, ergibt das Vektorprodukt t × w die Richtung der u-Achse.
5. Berechnung der v-Achse als Vektorprodukt der w- und u-Achsen: v = w × u.

In Animationen wird die Kameradefinition oft aus bestimmten Bedingungen automatisch berechnet, z.B. wenn die Kamera rund um ein Objekt fährt oder bei einer Flugsimulation, so dass gewünschte Effekte unkompliziert erreicht werden.

Um die Weltkoordinaten in Viewingkoordinaten umzurechnen braucht man eine Kette von einfachen Transformationen: zum Beispiel eine Translation der Koordinatenursprünge aufeinander und anschließend 3 Rotationen, so dass die Koordinatenachsen ebenfalls übereinander liegen (zwei Drehungen für die erste Achse, eine für die zweite Achse, und die dritte ist dann automatisch korrekt). Diese Transformationen kann man natürlich wieder durch Multiplikation zu einer Matrix zusammenfassen, das sieht etwa so aus:

$$\mathbf{M_{WC,VC} = R_z \cdot R_y \cdot R_x \cdot T}$$



Um die anschließende Projektion vollständig zu beschreiben, benötigt man noch die Grenzen des darzustellenden Bereiches. Zur vollständigen Definition der Kamera gehört demnach noch:

6. Die Wahl von minimalen und maximalen u-, v- und w-Werten zur Eingrenzung des Ausschnittes der Szene, der abgebildet wird: L(eft), R(ight), B(ottom), T(op), N(ear), F(ar).

Dabei beziehen sich L, R, B, T bei der Parallelprojektion auf die jeweiligen Werte der Grenzebenen (u=L, u=R, v=B, v=T), bei der perspektivischen Projektion auf die Bildgrenzen in der near-Ebene.

Orthographisches Viewing

Für die orthographische Projektion (Kamera bildet parallel ab) haben wir nun alle Schritte durch Matrizen beschrieben, diese können wir wie bei den geometrischen Transformationen zu einer einzigen Matrix zusammensetzen (multiplizieren), die dann die gesamte Viewing-Transformation durchführt:

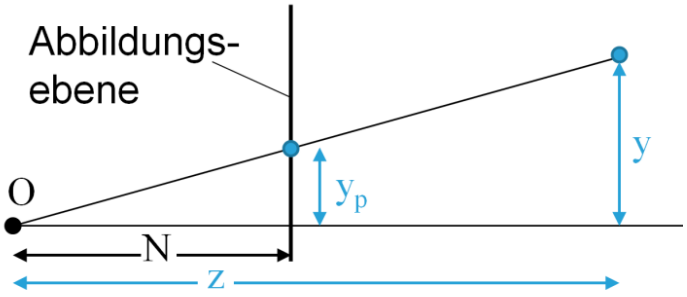
$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ z \\ 1 \end{bmatrix} = (M_{vp} \cdot M_{orth} \cdot M_{cam}) \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Man beachte: die rechteste Matrix wird zuerst mit dem Punkt (x,y,z) multipliziert. Wenn man das Assoziativgesetz anwendet, dann kann man aber auch zuerst die 3 Matrizen miteinander multiplizieren, und kann damit alle Punkte nur mit dieser einen Ergebnismatrix direkt von Weltkoordinaten in Gerätekoordinaten transformieren!

Perspektive

Bei der Perspektive gelten mehrere der Gesetze für affine Transformationen nicht mehr (z.B. verlaufen parallele Linien nach einer perspektivischen Projektion i.A. nicht mehr parallel), daher ist es keine affine Transformation und lässt sich auch nicht durch eine 3x3-Matrix erzeugen. Zum Glück helfen auch hier wieder die homogenen Koordinaten, allerdings ist dies der einzige Fall, bei dem die homogene Komponente h nicht den Wert 1 behält, und daher ein anschließender Divisionsschritt durch diesen Wert notwendig ist.

Zuerst wollen wir uns eine Formel für die perspektivische Transformation überlegen: Wir bezeichnen mit O das Projektionszentrum der Abbildung, und die Blickrichtung ist die negative z-Richtung. Dann befindet sich die Abbildungsebene normal auf die z-Achse im Abstand N(ear). Bildet man einen Punkt (x,y,z) auf diese Ebene ab, so hat er die Koordinaten (x·N/z, y·N/z, N). Dies lässt sich durch eine Matrix P bewerkstelligen:

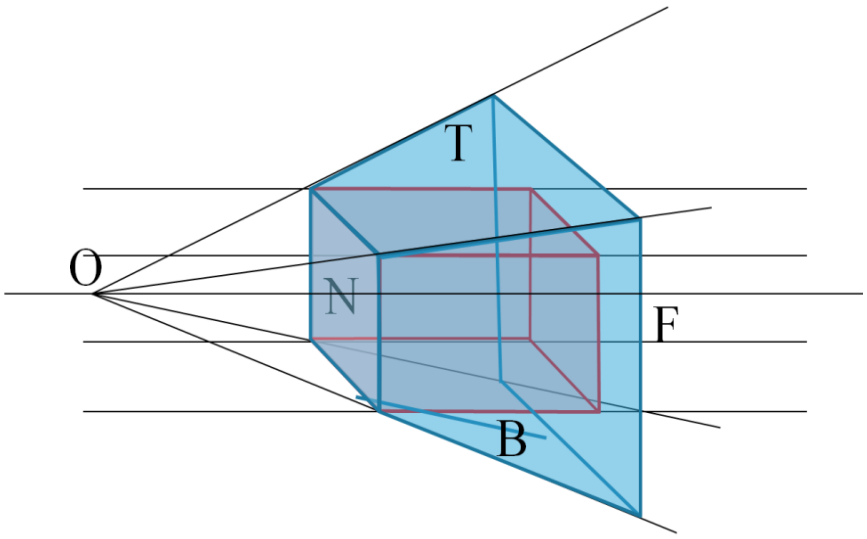


$$\mathbf{P} = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & N+F & -F \cdot N \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$y_p = \frac{N}{z} y \qquad x_p = \frac{N}{z} x$$

Multipliziert man einen Punkt (x,y,z,1) mit P so erhält man zuerst einmal (x·N, y·N, z·(N+F)–F·N, z). Durch *Homogenisieren* (auch Normalisieren = Dividieren durch die letzte Komponente, also z) entsteht

(x·N/z, y·N/z, (N+F)–F·N/z, 1).



Diese Operation entspricht einer Verzerrung des abzubildenden Szenebereiches (diesen Pyramidenstumpf nennt man „*View Frustum*“) in einen achsenparallelen Quader, in dem die orthographische Projektion genau dasselbe Bild liefert, wie die perspektivische Projektion im View Frustum. Anschließend können wir also die schon erarbeitete Parallelprojektion anwenden und damit die perspektivische Matrix M_{per} errechnen. Alternativ kann man auch P einfach an der richtigen Stelle der Gesamtviewingberechnung einsetzen und damit eine Gesamtmatrix erstellen, die von Modellkoordinaten (x,y,z) bis zu Gerätekoordinaten (Pixelpositionen (x_{screen}, y_{screen})) alles in einem Schritt ausführt:

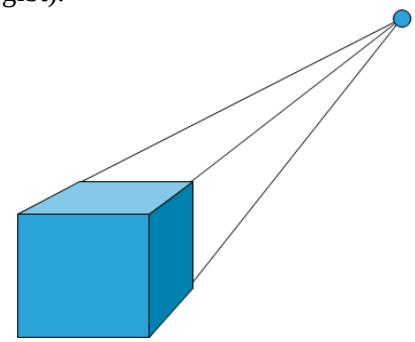
$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ z' \\ 1 \end{bmatrix} = M_{vp} \cdot \overbrace{(M_{orth} \cdot P \cdot M_{cam} \cdot M_{mod})}^{M_{per}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Wenn eine perspektivische Abbildung beteiligt ist, muss das Ergebnis am Ende durch die homogene Komponente (in diesem Fall z‘) dividiert werden. In der Praxis wird an der mit dem Werkzeug gekennzeichneten Stelle, also im Clipraum, nicht nur das Clipping durchgeführt, sondern auch die Homogenisierung. Danach wird die Viewport-Matrix als letzter Schritt getrennt angewendet.

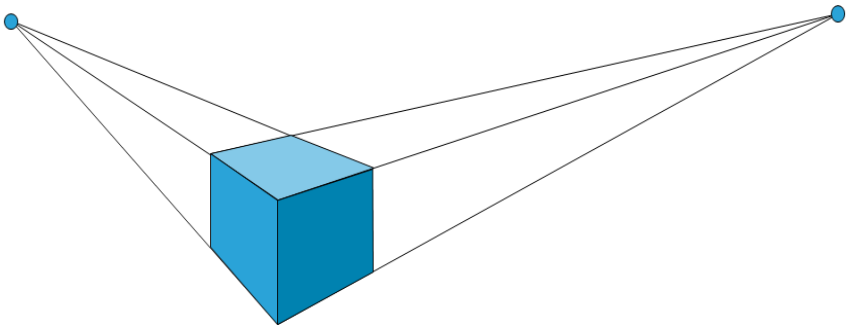
Wichtige Eigenschaften der Projektionstransformation sind weiters:
1. gerade Strecken bleiben gerade Strecken. Um eine solche Strecke (z.B. die Seite eines Polygons) abzubilden, reicht es, die beiden Endpunkte zu transformieren.
2. die relative Ordnung der z-Werte (also Abstand von der Kamera) bleibt erhalten (nicht aber die Abstandswerte selbst), das ist wichtig für die Sichtbarkeitsberechnung:

$z_1, z_2, N, F < 0$
$z_1 < z_2$
$1/z_1 > 1/z_2 \quad | \cdot (-F \cdot N) \quad (<0)$
$-F \cdot N/z_1 < -F \cdot N/z_2 \quad | + (N+F)$
$(N+F) - F \cdot N/z_1 < (N+F) - F \cdot N/z_2$

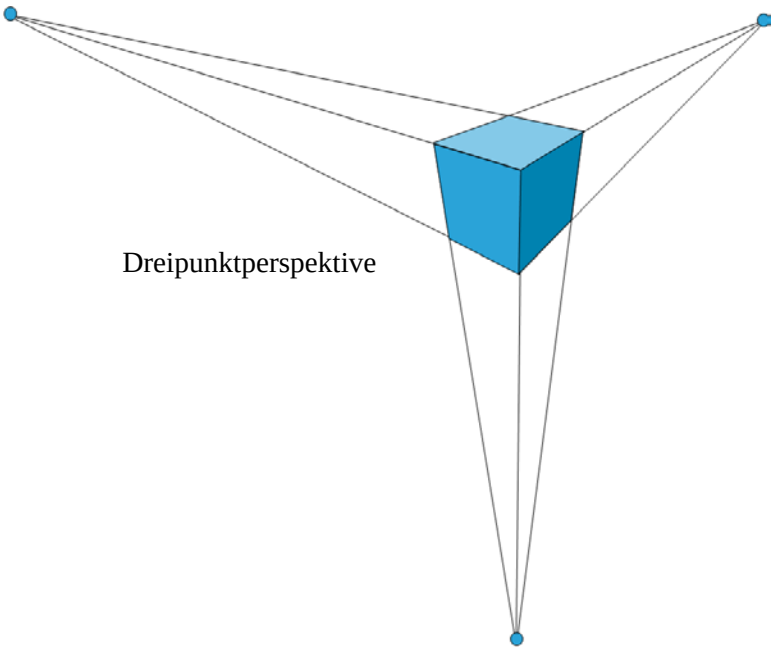
Abschließend sei noch erwähnt, dass die Anzahl der *Hauptfluchtpunkte* von der Lage der Bildebene zum Koordinatensystem abhängt. Wenn zwei Achsen parallel zur Bildebene liegen, dann spricht man von *Einpunktperspektive*, ist nur eine Achse parallel zur Bildebene, dann spricht man von *Zweipunktperspektive*, und sind alle drei Achsen nicht parallel zur Bildebene, dann spricht man von *Dreipunktperspektive* (weil es dann 3 Hauptfluchtpunkte gibt).



Einpunktperspektive



Zweipunktperspektive



Dreipunktperspektive